Package 'pmml'

July 2, 2014

Type Package

Title Generate PMML for various models

Version 1.4.2

Date 2013-10-10

Author Graham Williams, Tridivesh Jena, Wen Ching Lin, Michael Hahsler (arules),Zementis Inc, Hemant Ishwaran, Udaya B. Kogalur, Rajarshi Guha

Maintainer Tridivesh Jena <rpmmlsupport@zementis.net>

Depends XML

Suggests ada, amap, arules, glmnet, nnet, rattle, rpart,randomForestSRC, randomForest, kernlab, e1071, mlbench,pmmlTransformations

Imports survival

License GPL (>= 2)

Description The Predictive Model Markup Language (PMML) is an XML-based language which provides a way for applications to define statistical and data mining models and to share models between PMML compliant applications. More information about PMML and the Data Mining Group can be found at http://www.dmg.org. The generated PMML can be imported into any PMML consuming application,such as the Zementis ADAPA and UPPI scoring engines which allow for predictive models built in R to be deployed and executed on site, in the cloud (Amazon, IBM, and FICO), in-database (IBM Netezza, Pivotal,Sybase IQ, Teradata and Teradata Aster) or Hadoop (Datameer and Hive).

URL http://rattle.togaware.com/

Repository CRAN

NeedsCompilation no

Date/Publication 2014-06-17 08:24:33

R topics documented:

---

addLT *adds a LocalTransformations element to a given PMML file.*

Description

The pmmlTransformations package allows one to create a LocalTransformations element describing the data manipulations desired. This function allows one to add this information to a given PMML file; thus combining the description of any data processing as well as the model using such transformed data.

Usage

```
addLT(xmlmodel=NULL, transforms=NULL, namespace="4_2",...)
```

Arguments

xmlmodel	the PMML model in a XML node format. If the model is a text file, it should be converted to an XML node, for example, using the fileToXMLNode function.
transforms	the transformations performed on the initial data. This is the LocalTransformations element as an XML node object.
namespace	the namespace of the PMML model. This is frequently also the PMML version the model is represented as.
...	further arguments passed to or from other methods.

Details

The attribute information should be provided as a dataframe; each row corresponding to an attribute name and each column corresponding to a variable name. This way one can add as many attributes to as many variables as one wants in one step. On the other extreme, a one-by-one data frame may be used to add one new attribute to one variable. This function may be used multiple times to add new attribute values step-by-step. This function overwrites any pre-existing attribute values, so it must be used with care. However, this is by design as this feature is meant to help an user defined new attribute values at different times. For example, one may use this to impute missing values in a model at fifferent times.

Value

An object of class XMLNode as that defined by the **XML** package. This represents the top level, or root node, of the XML document and is of type PMML. It can be written to file with saveXML.

Author(s)

<tridivesh.jena@zementis.com>

Examples

```
## Not run:
# make a sample model

library(pmml)
model <- pmml(lm(Sepal.Length~., data=iris[,-5]))

# Perform a z-score transform on the first variable of the data set.
# As it is created and used in the same R session, this object is already an xml node,
# not an external text file; so there is no need to convert it to an xml node object.

library(pmmlTransformations)
irisBox <- WrapData(iris)
irisBox <- ZScoreXform(irisBox,"1")
xforms <- pmml(,transforms=irisBox)

# add the LocalTransformations element to the initial PMML model
# since the model still uses the original fields, the usage envisioned for this function
```

```
# is to make it easy if the modeller forgot to add the transformations when using the pmml
# function initially.

modified <- addLT(model,xforms,namespace="4_2")

## End(Not run)
```

`addMSAttributes` *adds attribute values to an existing MiningField element in a given PMML file*

Description

The PMML format allows a MiningField element to have attributes 'usageType', 'missingValueReplacement' and 'invalidValueTreatment' which although useful, may not always be present in a PMML model. This function allows one to take an existing PMML file and add these attributes to the MiningFields.

Usage

```
addMSAttributes(xmlmodel=NULL,attributes=NULL,namespace="4_2",...)
```

Arguments

`xmlmodel`	the PMML model in a XML node format. If the model is a text file, it should be converted to an XML node, for example, using the fileToXMLNode function.
`attributes`	the attributes to be added to the mining fields. The user should make sure that the attributes being added are allowed in the PMML schema.
`namespace`	the namespace of the PMML model. This is frequently also the PMML version the model is represented as.
`...`	further arguments passed to or from other methods.

Details

The attribute information should be provided as a dataframe; each row corresponding to an attribute name and each column corresponding to a variable name. This way one can add as many attributes to as many variables as one wants in one step. On the other extreme, a one-by-one data frame may be used to add one new attribute to one variable. This function may be used multiple times to add new attribute values step-by-step. This function overwrites any pre-existing attribute values, so it must be used with care. However, this is by design as this feature is meant to help an user defined new attribute values at different times. For example, one may use this to impute missing values in a model at fifferent times.

Value

An object of class `XMLNode` as that defined by the **XML** package. This represents the top level, or root node, of the XML document and is of type PMML. It can be written to file with `saveXML`.

Author(s)

<tridivesh.jena@zementis.com>

Examples

```
# make a sample model
library(pmml)
model0 <- lm(Sepal.Length~., data=iris[,-5])
model <- pmml(model0)

# Resulting model has mining fields with no information besides fieldName, dataType and optype.
# this object is already an xml node, not an external text file; so there is no need to convert
# it to an xml node object.

# create data frame with attribute information

attributes <- data.frame(c("active",1.1,"asIs"),c("active",2.2,"asIs"),c("active",NA,"asMissing"))
rownames(attributes) <- c("usageType","missingValueReplacement","invalidValueTreatment")
colnames(attributes) <- c("Sepal.Width","Petal.Length","Petal.Width")

#although not needed in this first try, necessary to easily add new values later
for(k in 1:ncol(attributes)){attributes[[k]]<-as.character(attributes[[k]])}

# actual command
addMSAttributes(model,attributes,namespace="4_2")
```

---

audit *Artificially constructed dataset*

Description

This is an artificial dataset consisting of fictional clients who have been audited, perhaps for tax refund compliance. For each case an outcome is recorded (whether the taxpayer's claims had to be adjusted or not) and any amount of adjustment that resulted is also recorded.

Format

A data frame containing:

Age	Numeric
Employment	Categorical string with 7 levels
Education	Categorical string with 16 levels
Marital	Categorical string with 6 levels
Occupation	Categorical string with 14 levels
Income	Numeric
Sex	Categorical string with 2 levels
Deductions	Numeric

Hours	Numeric
Accounts	Categorical string with 32 levels
Adjustment	Numeric
Adjusted	Numeric value 0 or 1

References

Togaware rattle package : ***Audit dataset***
http://www.dmg.org/pmml_examples/index.html#Audit

Examples

```
data(audit, package = "pmml")
```

`fileToXMLNode` *Reads in a file and tries to parse it into an object of type XMLNode*

Description

This function can be used when the user wants to read in an external file and convert it into an XMLNode to be used subsequently by other R functions.

Usage

```
fileToXMLNode(file)
```

Arguments

`file`	the external file to be read in. This file can be any file in PMML format, regardless of the source or model type.

Details

This function reads in a file and attempts to parse it into an XML node. This format is the one that will be obtained when a model is constructed in R and output in PMML format.

This function is mainly meant to be used to read in external files instead of depending on models saved in R. As an example, the pmml package requires as input an object of type XMLNode before its functions can be applied. Function 'fileToXMLNode' can be used to read in an existing PMML file, convert it to an XML node and then make it available for use by any of the pmml functions.

Value

An object of class `XMLNode` as that defined by the **XML** package. This represents the top level, or root node, of the XML document and is of type PMML. It can be written to file with `saveXML`.

Author(s)

<tridivesh.jena@zementis.com>

Examples

```
# define some transformations
 library(pmml)
 library(pmmlTransformations)

 irisBox <- WrapData(iris)
 irisBox <- ZScoreXform(irisBox,xformInfo = "column1->d1")
 irisBox <- ZScoreXform(irisBox,xformInfo = "column2->d2")

#make a LocalTransformations element and save it to an external file
 pmml_trans <- pmml(NULL, transforms=irisBox)
 write(toString(pmml_trans),file = "xform_iris.pmml")

 # Later, we may need to read in the PMML model into R
 # 'lt' below is now a XML Node, as opposed to a string

 lt <- fileToXMLNode("xform_iris.pmml")
```

---

pmml *Generate PMML for R objects*

Description

pmml is a generic function implementing S3 methods used to produce the PMML (Predictive Model Markup Language) representation of an R model. The resulting PMML file can then be imported into other systems that accept PMML.

The same function can also be used to output variable transformations in PMML format. In particular, it can be used as a transformations generator. Various transformation operations can be implemented in R and those transformations can then be output in PMML format by calling the function with a NULL value for the model input and a pmmlTransformations object as the transforms input. Please see the R **pmmlTransformations** package for more information on how to create the pmmlTransformations object.

In addition, the pmml function can also be called using a pre-existing PMML model as the first input and a pmmlTransformations object as the transforms input. The result is a new PMML model with the transformation inserted as a "LocalTransformations" element in the original model. If the original model already had a "LocalTransformations" element, the new information will be appended to that element. If the model variables are derived directly from a chain of transformations defined in the transforms input, the field names in the model are replaced with the original field names with the correct data types to make a consistent model. The covered cases include model fields derived from an original field, model fields derived from a chain of transformations starting from an original field and mutiple fields derived from the same original field.

Please note that package **XML_3.95-0.1** or later is required to perform the full and correct functionality of the **pmml** package.

Usage

```
pmml(model=NULL, model.name="Rattle_Model", app.name="Rattle/PMML",
     description=NULL, copyright=NULL, transforms=NULL, ...)
```

Arguments

`model`	an object to be converted to PMML.
`model.name`	a name to be given to the model in the PMML code.
`app.name`	the name of the application that generated the PMML code.
`description`	a descriptive text for the Header element of the PMML code.
`copyright`	the copyright notice for the model.
`transforms`	data transformations represented in PMML via package **pmmlTransformations**.
`...`	further arguments passed to or from other methods.

Details

PMML is an XML based language which provides a way for applications to define statistical and data mining models and to share models between PMML compliant applications. More information about PMML and the Data Mining Group can be found at http://www.dmg.org.

The generated PMML can be imported into any PMML consuming application, such as the Zementis ADAPA and UPPI scoring engines which allow for predictive models built in R to be deployed and executed on site, in the cloud (Amazon, IBM, and FICO), in-database (IBM Netezza, Pivotal, Sybase IQ, Teradata and Teradata Aster) or Hadoop (Datameer and Hive).

Value

An object of class `XMLNode` as that defined by the **XML** package. This represents the top level, or root node, of the XML document and is of type PMML. It can be written to file with `saveXML`.

Author(s)

<Graham.Williams@togaware.com>

References

- Rattle home page: http://rattle.togaware.com
- PMML home page: http://www.dmg.org
- A. Guazzelli, W. Lin, T. Jena (2012), *PMML in Action: Unleashing the Power of Open Standards for Data Mining and Predictive Analytics*. CreativeSpace (Second Edition) - Available on Amazon.com http://www.amazon.com/dp/1470003244.
- A. Guazzelli, M. Zeller, W. Lin, G. Williams (2009), PMML: An Open Standard for Sharing Models. *The R journal*, Volume 1/1, 60-65

- A. Guazzelli, T. Jena, W. Lin, M. Zeller (2013). Extending the Naive Bayes Model Element in PMML: Adding Support for Continuous Input Variables. In *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* http://kdd13pmml.files.wordpress.com/2013/07/guazzelli_et_al.pdf
- T. Jena, A. Guazzelli, W. Lin, M. Zeller (2013). The R pmmlTransformations Package. In *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* http://kdd13pmml.files.wordpress.com/2013/07/jena_et_al.pdf

See Also

pmml.ada, pmml.rules, pmml.coxph, pmml.cv.glmnet, pmml.glm, pmml.hclust, pmml.kmeans, pmml.ksvm, pmml.lm, pmml.multinom, pmml.naiveBayes, pmml.nnet, pmml.randomForest, pmml.rfsrc, pmml.rpart, pmml.svm

Examples

```
# Build a simple lm model
iris.lm <- lm(Sepal.Length ~ ., data=iris)

# Convert to pmml
pmml(iris.lm)

# Create a pmmlTransformations object
library(pmmlTransformations)
xo <- WrapData(iris)

# Transform the 'Sepal.Length' variable
xo <- MinMaxXform(xo,xformInfo="column1->d_sl")

# Output the tranformation in PMML format
pmml(NULL, transforms=xo)
```

---

pmml.ada *Generate PMML for ada objects*

Description

Generate the PMML representation for an ada object from package **ada**.

Usage

```
## S3 method for class 'ada'
pmml(model, model.name="AdaBoost_Model", app.name="R-PMML",
     description="AdaBoost Model", copyright=NULL, transforms=NULL,
     unknownValue=NULL, ...)
```

Arguments

model	ada object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
...	further arguments passed to or from other methods.

Details

The pmml function exports the ada model in the PMML MiningModel (multiple models) format. The MiningModel element consists of a list of TreeModel elements, one in each model segment.

Note that each segment tree is a classification model, returning either -1 or 1. However the MiningModel (ada algorithm) is doing a weighted sum of the returned value, -1 or 1. So the value of attribute functionName of element MiningModel is set to "regression"; the value of attribute functionName of each segment tree is also set to "regression" (they have to be the same as the parent MiningModel per PMML schema). Although each segment/tree is being named a "regression" tree, the actual returned score can only be -1 or 1, which practically turns each segment into a classification tree.

The model in PMML format has 5 different outputs. The "rawValue" output is the value of the model expressed as a tree model. The boosted tree model uses a transformation of this value, this is the "boostValue" output. The last 3 outputs are the predicted class and the probabilities of each of the 2 classes (The ada package Boosted Tree models can only handle binary classification models).

Author(s)

Zementis Inc. <info@zementis.com>

References

R project CRAN package: ***ada****: an R package for stochastic boosting*
http://cran.r-project.org/web/packages/ada/index.html

Examples

```
library(ada)
library(pmml)
data(audit)

fit <- ada(Adjusted~Employment+Education+Hours+Income,iter=3, audit)
pmml_fit <- pmml(fit)
```

pmml.coxph *Generate PMML for coxph objects*

Description

Generate the PMML representation for a coxph object from package **survival**.

Usage

```
## S3 method for class 'coxph'
pmml(model, model.name="CoxPH_Survival_Regression_Model", app.name="Rattle/PMML",
    description="CoxPH Survival Regression Model", copyright=NULL, transforms=NULL,
     unknownValue=NULL, ...)
```

Arguments

model	a coxph object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
...	further arguments passed to or from other methods.

Details

A coxph object is the result of fitting a proportional hazards regression model, using the "coxph" function from the package **survival**. Although the **survival** package supports special terms "cluster", "tt" and "strata", only the special term "strata" is supported by the **pmml** package. Note that special term "strata" cannot be a multiplicative variable and only numeric risk regression is supported.

Author(s)

<Graham.Williams@togaware.com>, Zementis Inc. <info@zementis.com>

References

R project CRAN package: ***survival**: Survival Analysis*
http://cran.r-project.org/web/packages/survival/index.html

`pmml.cv.glmnet` *Generate PMML for glmnet objects*

Description

Generate the PMML representation for a glmnet (elasticnet general linear regression) object. In particular, this gives the PMML representation for an object created by the cv.glmnet function.

Usage

```
## S3 method for class 'cv.glmnet'
pmml(model, model.name="Elasticnet_Model", app.name="Rattle/PMML",
     description="Generalized Linear Regression Model", copyright=NULL,
     transforms=NULL, unknownValue=NULL, dataset=NULL, s=NULL, ...)
```

Arguments

`model`	a `cv.glmnet` object contained in an object of class **glmnet**, as contained in the object returned by the function `cv.glmnet`.
`model.name`	a name to be given to the model in the PMML code.
`app.name`	the name of the application that generated the PMML code.
`description`	a descriptive text for the Header element of the PMML code.
`copyright`	the copyright notice for the model.
`transforms`	data transformations represented in PMML via package **pmmlTransformations**.
`unknownValue`	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
`dataset`	the dataset using which the model was built.
`s`	the 'lambda' parameter at which to output the model. If not given, the lambda.min parameter from the model is used instead.
`...`	further arguments passed to or from other methods.

Details

The `glmnet` package expects the input and predicted values in a matrix format; not as arrays or data frames. As of now, it will also accept numerical values only. As such, any string variables must be converted to numerical ones. One possible way to do so is to use data transformation functions, such as from the **pmmlTransformations** package. However the result is a data frame. In all cases, lists, arrays and data frames can be converted to a matrix format using the data.matrix function from the base package. Given a data frame df, a matrix m can thus be created by using m <- data.matrix(df).

The PMML language requires variable names which will be read in as the column names of the input matrix. If the matrix does not have variable names, they will be given the default values of "X1", "X2", ...

Use of PMML and `pmml.cv.glmnet` requires the **XML** package. Be aware that XML is a very verbose data format.

Author(s)

Zementis Inc. <info@zementis.com>

References

R project CRAN package:
***glmnet**: Lasso and elastic-net regularized generalized linear models*
http://cran.r-project.org/web/packages/glmnet/index.html

Examples

```
library(glmnet)

# create a simple predictor (x) and response(y) matrices
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)

# Build a simple gaussian model
model1 = cv.glmnet(x,y)
# Output the model in PMML format
pmml(model1)

# shift y between 0 and 1 to create a poisson response
y = y - min(y)
# give the predictor variables names (default values are V1,V2,...)
name <- NULL
for(i in 1:20){
  name <- c(name,paste("variable",i,sep=""))
}
colnames(x) <- name
# create a simple poisson model
model2 <- cv.glmnet(x,y,family="poisson")
# output in PMML format the regression model at the lambda parameter = 0.006
pmml(model2,s=0.006)
```

---

pmml.glm *Generate PMML for glm objects*

Description

Generate the PMML representation for a glm object from package **stats**.

Usage

```
## S3 method for class 'glm'
pmml(model, model.name="General_Regression_Model", app.name="Rattle/PMML",
     description="Generalized Linear Regression Model", copyright=NULL,
     transforms=NULL, unknownValue=NULL, weights=NULL, ...)
```

Arguments

model	a glm object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
weights	the weights used for building the model.
...	further arguments passed to or from other methods.

Details

The function exports the glm model in the PMML GeneralRegressionModel format.

Author(s)

Zementis Inc. <info@zementis.com>

References

R project: ***Fitting Generalized Linear Models***
http://stat.ethz.ch/R-manual/R-devel/library/stats/html/glm.html

---

pmml.hclust *Generate PMML for hclust objects*

Description

Generate the PMML representation for a hierarchical cluster object. The hclust object will be approximated by k centroids and is converted into a PMML representation for kmeans clusters.

Usage

```
## S3 method for class 'hclust'
pmml(model, model.name="HClust_Model", app.name="Rattle/PMML",
     description="Hierarchical cluster model", copyright=NULL,
     transforms=NULL, unknownValue=NULL, centers, ...)
```

Arguments

model	a hclust object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
centers	a list of means to represent the clusters.
...	further arguments passed to or from other methods.

Details

This function converts a hclust object created by the 'hclusterpar' function from the 'amap' package. A hclust object is a cluster model created hierarchically. The data is divided recursively until a criteria is met. This function then takes the final model and represents it as a standard k-means cluster model. This is possible since while the method of constructing the model is different, the final model can be represented in the same way.

To use this pmml function, therefore, one must pick the number of clusters desired and the coordinate values at those cluster centers. This can be done using the 'hclusterpar' and 'centers.hclust' functions from the 'amap' and 'rattle' packages repectively.

Author(s)

<Graham.Williams@togaware.com>

References

R project: ***Hierarchical Clustering***
http://stat.ethz.ch/R-manual/R-devel/library/stats/html/hclust.html

Examples

```
#\dontrun{
# cluster the 4 numeric variables of the iris dataset
library(amap)
model <- hclusterpar(iris[,-5])

# get the information about the cluster centers
# the last parameter of the function used is the number of clusters desired
 library(rattle)
centerInfo <- centers.hclust(iris[,-5],model,3)

# convert to pmml
library(pmml)
```

```
pmml(model,centers=centerInfo)
#}
```

---

pmml.kmeans *Generate PMML for kmeans objects*

Description

Generate the PMML representation for a kmeans object (cluster) from package **stats**. The kmeans object (a cluster described by k centroids) is converted into a PMML representation.

Usage

```
## S3 method for class 'kmeans'
pmml(model, model.name="KMeans_Model", app.name="Rattle/PMML",
     description="KMeans cluster model", copyright=NULL,
    transforms=NULL, unknownValue=NULL, algorithm.name="KMeans: Hartigan and Wong", ...)
```

Arguments

model	a kmeans object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
algorithm.name	the variety of kmeans used.
...	further arguments passed to or from other methods.

Details

A kmeans object is obtained by applying the kmeans function from the stats package. This method typically requires the user to normalize all the variables, these operations can be done using the pmmlTransformations package so that the normalization information is included in the pmml model format.

Author(s)

<Graham.Williams@togaware.com>

References

R project: ***K-Means Clustering***
http://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html

Examples

```
ds <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
                matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
colnames(ds) <- c("Dimension1", "Dimension2")
cl <- kmeans(ds, 2)
pmml(cl)
```

`pmml.ksvm` *Generate PMML for ksvm objects*

Description

Generate the PMML representation for a ksvm object from package **kernlab**.

Usage

```
## S3 method for class 'ksvm'
pmml(model, model.name="SVM_model", app.name="Rattle/PMML",
     description="Support Vector Machine PMML Model", copyright=NULL,
     transforms=NULL, unknownValue=NULL, dataset=NULL, ...)
```

Arguments

`model`	a ksvm object.
`model.name`	a name to be given to the model in the PMML code.
`app.name`	the name of the application that generated the PMML code.
`description`	a descriptive text for the Header element of the PMML code.
`copyright`	the copyright notice for the model.
`transforms`	data transformations represented in PMML via package **pmmlTransformations**.
`unknownValue`	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
`dataset`	required since the ksvm object does not record information about the used categorical variable; the original dataset used to train the SVM model in ksvm.
`...`	further arguments passed to or from other methods.

Details

Both classification (multi-class and binary) as well as regression cases are supported.

Author(s)

Zementis Inc. <info@zementis.com>

References

R project CRAN package: ***kernlab****: Kernel-based Machine Learning Lab*
http://cran.r-project.org/web/packages/kernlab/index.html

Examples

```
# Train a support vector machine to perform classification.
library(kernlab)
model  <- ksvm(Species ~ ., data=iris)
p <- pmml(model, dataset=iris)

# To make predictions using this model, the new data must be given; without it and by
# simply using the "predict" function without an input dataset, the predicted value
# will not be the true predicted value. It will be a raw predicted value which must be
# post-processed to get the final correct predicted value
#
# Make predictions using same iris input data. Even though it is the same dataset, it
# must be provided as an input parameter for the "predict" function.

predict(model,iris[,1:4])

rm(model)
rm(p)
```

---

pmml.lm *Generate PMML for lm objects*

Description

Generate the PMML representation for a lm object from package **stats**.

Usage

```
## S3 method for class 'lm'
pmml(model, model.name="Linear_Regression_Model", app.name="Rattle/PMML",
     description="Linear Regression Model", copyright=NULL,
     transforms=NULL, unknownValue=NULL, dataset=NULL, weights=NULL, ...)
```

Arguments

model	a lm object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.

description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
dataset	the orginal training dataset, if available.
weights	the weights used for building the model.
...	further arguments passed to or from other methods.

Details

Note that the resulting PMML representation will not encode interaction terms. Currently, only numeric regression is supported.

Author(s)

<rguha@indiana.edu>

References

R project: ***Fitting Linear Models***
http://stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html

Examples

```
fit <- lm(Sepal.Length ~ ., data=iris)
pmml(fit)

rm(fit)
```

---

pmml.multinom *Generate PMML for multinom objects*

Description

Generate the PMML representation for a multinom object from package **nnet**.

Usage

```
## S3 method for class 'multinom'
pmml(model, model.name="multinom_Model", app.name="Rattle/PMML",
     description="Multinomial Logistic Model", copyright = NULL,
     transforms = NULL, unknownValue=NULL, ...)
```

Arguments

model	a multinom object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
...	further arguments passed to or from other methods.

Details

This function outputs the multinomial logistic model in the PMML RegressionModel format. It implements the use of numerical, categorical and multiplicative terms involving both numerical and categorical variables.

Author(s)

Zementis Inc. <info@zementis.com>

References

R project CRAN package:
***nnet**: Feed-forward Neural Networks and Multinomial Log-Linear Models*
http://cran.r-project.org/web/packages/nnet/index.html

---

pmml.naiveBayes *Generate PMML for naiveBayes objects*

Description

Generate the PMML representation for a naiveBayes object from package **e1071**.

Usage

```
## S3 method for class 'naiveBayes'
pmml(model, model.name="naiveBayes_Model", app.name="Rattle/PMML",
    description="NaiveBayes Model", copyright=NULL, transforms=NULL, unknownValue=NULL,
     predictedField, ...)
```

Arguments

model	a naiveBayes object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
predictedField	Required parameter; the name of the predicted field.
...	further arguments passed to or from other methods.

Details

The PMML representation of the NaiveBayes model implements the definition as specified by the Data Mining Group: intermediate probability values which are less than the threshold value are replaced by the threshold value. This is different from the prediction function of the **e1071** in which only probability values of 0 and standard deviations of continuous variables of with the value 0 are replaced by the threshold value. The two values will therefore not match exactly for cases involving very small probabilities.

Author(s)

Zementis Inc. <info@zementis.com>

References

- R project CRAN package:
  ***e1071**: Misc Functions of the Department of Statistics (e1071), TU Wien*
  http://cran.r-project.org/web/packages/e1071/index.html
- A. Guazzelli, T. Jena, W. Lin, M. Zeller (2013). Extending the Naive Bayes Model Element in PMML: Adding Support for Continuous Input Variables. In *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Examples

```
# Build a simple Naive Bayes model

# Upload the required library
library(e1071)
library(pmml)
library(mlbench)

# download an example dataset
data(HouseVotes84)
house <- na.omit(HouseVotes84)
```

```
# Construct an example model defining a threshold value of 0.003
model<-naiveBayes(Class~V1+V2+V3,data=house,threshold=0.003)

# Output the PMML representation
pmml(model,dataset=house,predictedField="Class")

rm(model)
```

---

pmml.nnet *Generate PMML for nnet objects*

Description

Generate the PMML representation for a nnet object from package **nnet**.

Usage

```
## S3 method for class 'nnet'
pmml(model, model.name="NeuralNet_model", app.name="Rattle/PMML",
     description="Neural Network PMML Model", copyright=NULL, transforms=NULL,
     unknownValue=NULL, ...)
```

Arguments

model	a nnet object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
...	further arguments passed to or from other methods.

Details

The pmml function supports both regression and classification neural network models. The model is represented in the PMML NeuralNetwork format.

Author(s)

Zementis Inc. <info@zementis.com>

References

R project CRAN package:
***nnet**: Feed-forward Neural Networks and Multinomial Log-Linear Models*
http://cran.r-project.org/web/packages/nnet/index.html

Examples

```
library(nnet)
fit <- nnet(Species ~ ., data=iris, size=4)
pmml(fit)

rm(fit)
```

`pmml.randomForest` *Generate PMML for randomForest objects*

Description

Generate the PMML representation for a randomForest object from package **randomForest**.

Usage

```
## S3 method for class 'randomForest'
pmml(model, model.name="randomForest_Model", app.name="Rattle/PMML",
     description="Random Forest Tree Model", copyright=NULL, transforms=NULL,
     unknownValue=NULL,  ...)
```

Arguments

`model`	a `randomForest` object.
`model.name`	a name to be given to the model in the PMML code.
`app.name`	the name of the application that generated the PMML code.
`description`	a descriptive text for the Header element of the PMML code.
`copyright`	the copyright notice for the model.
`transforms`	data transformations represented in PMML via package **pmmlTransformations**.
`unknownValue`	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
`...`	further arguments passed to or from other methods.

Details

This function outputs a Random Forest in PMML format. The model will include not just the forest but also any pre-processing applied to the training data.

Author(s)

Zementis Inc. <info@zementis.com>

References

R project CRAN package:
***randomForest**: Breiman and Cutler's random forests for classification and regression*
http://cran.r-project.org/web/packages/randomForest/index.html

Examples

```
# Build a simple randomForest model

library(randomForest)
iris.rf <- randomForest(Species ~ ., data=iris, ntree=20)

# Convert to pmml

pmml(iris.rf)

rm(iris.rf)
```

`pmml.rfsrc` *Generate PMML for rsf objects*

Description

Generate the PMML representation for a **randomSurvivalForest** forest object.

Usage

```
## S3 method for class 'rfsrc'
pmml(model, model.name="rsf_Model", app.name="Rattle/PMML",
    description="Random Survival Forest Model", copyright=NULL, transforms=NULL,
     unknownValue=NULL, ...)
```

Arguments

`model`	a `forest` object contained in an object of class **randomSurvivalForest**, as that contained in the object returned by the function `rsf` with the parameter "forest=TRUE".
`model.name`	a name to be given to the model in the PMML code.
`app.name`	the name of the application that generated the PMML code.
`description`	a descriptive text for the Header of the PMML code.
`copyright`	the copyright notice for the model.

transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
...	further arguments passed to or from other methods.

Details

This function is used to export the geometry of the forest to other PMML compliant applications, including graphics packages that are capable of printing binary trees. In addition, the user may wish to save the geometry of the forest for later retrieval and prediction on new data sets using pmml.rfsrc together with pmml_to_rsf.

Author(s)

Zementis Inc. <info@zementis.com>

References

- H. Ishwaran, U.B. Kogalur, E.H. Blackstone, M.S. Lauer (2008), /emphRANDOM SURVIVAL FORESTS. The Annals of Applied Statistics, Vol. 2, No. 3, 841-860
- H. Ishwaran and Udaya B. Kogalur (2006). Random Survival Forests. *Cleveland Clinic Technical Report*.

Examples

```
library(randomForestSRC)
data(veteran)
veteran.out <- rfsrc(Surv(time, status)~., data = veteran, ntree = 5, forest = TRUE)
pmml(veteran.out)
```

---

pmml.rpart *Generate PMML for rpart objects*

Description

Generate the PMML representation for a rpart object from package **rpart**.

Usage

```
## S3 method for class 'rpart'
pmml(model, model.name="RPart_Model", app.name="Rattle/PMML",
     description="RPart Decision Tree Model", copyright=NULL, transforms=NULL,
     unknownValue=NULL, dataset=NULL, ...)
```

Arguments

model	a rpart object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
dataset	the original dataset used to train the model.
...	further arguments passed to or from other methods.

Details

The pmml function supports regression tree as well as classification tree of a rpart object. The object is represented in the PMML TreeModel format.

Author(s)

<Graham.Williams@togaware.com>, Zementis Inc. <info@zementis.com>

References

R project CRAN package: ***rpart****: Recursive Partitioning*
http://cran.r-project.org/web/packages/rpart/index.html

Examples

```
library(rpart)
fit <- rpart(Species ~ ., data=iris)
pmml(fit)

rm(fit)
```

---

pmml.rules *Generate PMML for arules objects*

Description

Generate the PMML representation for a rules or an itemset object from package **arules**.

Usage

```
## S3 method for class 'rules'
pmml(model, model.name="arules_Model", app.name="Rattle/PMML",
    description="arules association rules model", copyright=NULL, transforms = NULL, ...)
## S3 method for class 'itemsets'
pmml(model, model.name="arules_Model", app.name="Rattle/PMML",
    description="arules frequent itemsets model", copyright=NULL, transforms = NULL, ...)
```

Arguments

model	a rules or itemsets object.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML code.
copyright	the copyright notice for the model.
transforms	not used in present version.
...	further arguments passed to or from other methods.

Details

The model is represented in the PMML AssociationModel format.

Author(s)

Michael Hahsler (<michael@hahsler.net>)

References

R project CRAN package: ***arules**: Mining Association Rules and Frequent Itemsets*
http://cran.r-project.org/web/packages/arules/index.html

---

pmml.svm *Generate PMML for svm objects*

Description

Generate the PMML representation of a svm object from the **e1071** package.

Usage

```
## S3 method for class 'svm'
pmml(model, model.name="LIBSVM_Model", app.name="R-PMML",
    description="Support Vector Machine Model", copyright=NULL, transforms=NULL,
     unknownValue=NULL, ...)
```

Arguments

model	a svm object from package **e1071**.
model.name	a name to be given to the model in the PMML code.
app.name	the name of the application that generated the PMML code.
description	a descriptive text for the Header element of the PMML.
copyright	the copyright notice for the model.
transforms	data transformations represented in PMML via package **pmmlTransformations**.
unknownValue	value to be used as the 'missingValueReplacement' attribute for all MiningFields.
...	further arguments passed to or from other methods.

Details

The model is represented in the PMML SupportVectorMachineModel format.

Note that the sign of the coefficient of each support vector flips between the R object and the exported PMML file. This is due to the minor difference in the training/scoring formula between the LIBSVM algorithm and the DMG specification. Hence the output value of each support vector machine has a sign flip between the DMG definition and the svm prediction function.

In a classification model, even though the output of the support vector machine has a sign flip, it does not affect the final predicted category. This is because in the DMG definition, the winning category is defined as the left side of threshold 0 while the LIBSVM defines the winning category as the right side of threshold 0.

For a regression model, the exported PMML code has two OutputField elements. The first OutputField "predictedValue" shows the support vector machine output per DMG definition. The second one "svm_predict_function" gives the value corresponding to the R predict function for the svm model. This output should be the one to use when making model predictions.

Author(s)

Zementis Inc. <info@zementis.com>

References

- R project CRAN package:
  ***e1071**: Misc Functions of the Department of Statistics (e1071), TU Wien*
  http://cran.r-project.org/web/packages/e1071/index.html
- Chang, Chih-Chung and Lin, Chih-Jen, *LIBSVM: a library for Support Vector Machines*
  http://www.csie.ntu.edu.tw/~cjlin/libsvm

Examples

```
library(e1071)
fit <- svm(Species ~ ., data=iris, kernel="polynomial")
pmml(fit)
```

```
rm(fit)
```

---

pmmlCanExport *Can this installation export PMML variables (particularly transforms).*

Description

This function is designed to be overriden by other packages that implement PMML export, particularly of transformations.

Usage

```
pmmlCanExport(vname)
```

Arguments

vname	a variable name to check whether it is exportable.

Author(s)

<Graham.Williams@togaware.com>

See Also

pmml.

---

pmmltoc *Generate C code from a PMML object - dummy function*

Description

This is a dummy function that does nothing. Plugins for Rattle are starting to appear which implement this for specific environments. This is experimental.

Usage

```
pmmltoc(p, name=NULL, includePMML=TRUE, includeMetaData=TRUE, exportClass=TRUE)
```

Arguments

p	pmml.
name	a name to give to the model in the C code.
includePMML	include the actual PMML as comments.
includeMetaData	include model information as comments.
exportClass	whether to export class or probability.

Author(s)

<Graham.Williams@togaware.com>

See Also

pmml.

Index